Package 'ANN'

February 15, 2013

Type Package

Title Feedforward Artificial Neural Network optimized by Genetic Algorithm

Version 0.1.4

Date 2012-04-06

Author Francis Roy-Desrosiers

Maintainer Francis Roy-Desrosiers <francis.roy.desrosiers@gmail.com>

Description This package provides a feedforward Artificial Neural
Network (ANN) optimized by Genetic Algorithm (GA).
Modifications in the Genetic algorithm from CUDAANN r6 have
been made to improve the speed, the convergence and the memory management.

Depends R (>= 2.12.0)

License GPL-3

LazyLoad yes

Repository CRAN

Repository/R-Forge/Project ann

Repository/R-Forge/Revision 5

Date/Publication 2012-04-21 04:25:14

NeedsCompilation yes

R topics documented:

ANN-package *Artificial Neural Network optimized by Genetic Algorithm*

Description

This package provides a feedforward Artificial Neural Network (ANN) optimized by Genetic Algorithm (GA). Modifications in the Genetic Algorithm from CUDAANN r6 have been made to improve the speed, the convergence and the memory management.

Details

Package:	ANN
Type:	Package
Version:	0.1.4
Date:	2012-04-06
License:	GPL-3
LazyLoad:	yes

An artificial neural network is based on the mathematical representation of a neural network or the human brain works. The artificial neural network (ANN) has neurons and synapses, similar to the neural network. The synapses are the connections between the neurons which are represented as weight (w_i). There are several connection types that exist to solve different kinds of problems. In this package, the feedforward artificial neural network is used. In this type of network, the information is gathered into the input layer which forwards the information through the network to the output layer.

The ANN consists of different layers. The input layer takes the input data, then distributes it to the connections that connect the hidden layer(s) and the input layer. The neuron(s) in the hidden layer(s) process(es) the summation of the information received from the connections of the input layer. Then it processes the summations with its activation function and distributes the result to the next layer. This process continues down through the layers to the output layer. The neuron(s) of the output layer process(es) the summation of the information received from the connections of the hidden layer. Then each neuron processes the summation with its activation function. The output of the activation function is the output of the ANN.

In this package, the activation function is sigmoid: $f(x) = \frac{1}{1+e^{-x}}$. An example of a formula for a 3-layer feedforward ANN (the input layer, one hidden layer and the output layer) with one output would be:

$f(\sum_{j=1}^{N_{hidden\ layer}} w_{2,j,1} f(\sum_{i=1}^{N_{input\ layer}} w_{1,i,j} input_i + bias_{1,j}) + bias_{2,1}) = output$

$w_{k,i,j}$, means weight from the ith neuron on the kth layer to the jth neuron on the next layer. The $bias_{k,j}$ is similar to a constant in a least square regression. Each neuron of the hidden layer and of the output layer has a bias and the neurons add the bias to the summation. The weights of the connections must be estimated. The learning phase of the ANN can be supervised or unsupervised. In this package supervised learning has been used. Supervised learning consists of giving inputs to the ANN and adjusting the weight to minimize the sum of the differences between the predicted

output given by the ANN and the desired output. The mean squared error $MSE = \frac{1}{N}\sum(y_i^{output} - y_i^{estimated\ output})^2$ is the criterion to be minimized in this package.

Several methods of minimisation are available. In this package, the Genetic Algorithm is used to solve the problem of estimating the connections' weights. Genetic Algorithm is based on natural evolution. A population of chromosomes is initialised with random numbers. In our case a chromosome is an array of doubles corresponding to weights. This population of chromosomes evolves within every generation; mutation and crossover occur in all chromosomes. Mutation ($w_{j,i}^{new} = w_{k,i} + mutationRate * (w_{n,j} - w_{m,i})$) consists of adding a weight i from a random chromosome to the difference of the weight i of two other random chromosomes multiplied by the mutation rate, where $w_{k,i}$ is the ith weight of the kth chromosome and k, n, m are randomly choosen within the set of chromosomes population and k, n, m are not equal. The crossover consists of changing each chromosome's weights to the mutation weight with the probability of the crossover rate. Then, it replaces the old chromosome with the new one if the MSE of the new chromosome is smaller. This algorithm was originaly developed by Emre Caglar in "CUDAANN r6 project".

Note

Comments, adjustments and/or contributions are welcome.

Author(s)

Francis Roy-Desrosiers

Maintainer: Francis Roy-Desrosiers <francis.roy.desrosiers at gmail.com>

References

Cross Ss, Harrison Rf, Kennedy RL.. Introduction to neural networks. Lancet 1995; 346: 1075-1079.

`ANN` *Printing/plot an ANN object*

Description

Printing/plot method for an ANN objects

Usage

```
## S3 method for class 'ANN'
print(x,...)
## S3 method for class 'ANN'
plot(x,...)
```

Arguments

`x`	Object of class "ANN", an already optimized ANN.
`...`	not used

Author(s)

Francis Roy-Desrosiesrs

ANNGA *Optimize an ANN by GA*

Description

This function uses a feedforward Artificial Neural Network optimized by Genetic Algorithm to minimize the mean squared error.

Usage

```
ANNGA ( x,
y,
design = c(1, 3, 1),
population = 500,
mutation = 0.3,
crossover = 0.7,
maxW = 25,
minW = -25,
maxGen = 1000,
error = 0.05,
printBestChromosone=TRUE,
...)
```

Arguments

x	A matrix of inputs, where each column is a different input. Value must be in the range [0,1].
y	A matrix of outputs, where each column is a different output. Value must be in the range [0,1].
design	A vector for the configuration of the network: the first number will be the number of inputs, the last number will be the number of outputs and the numbers in between are the numbers of neurons on each hidden layer. Example 1:design = c(1, 3, 1) one input, one hidden layer with three neurons and one neuron in the output layer. Example 2: design = c(5, 3, 2, 1) five inputs, two hidden layers, the first one with three neurons, the second one with two neurons and one neuron in the output layer.
population	The number of chromosomes in the population of each generation.
mutation	The mutation rate: should be between 0 and 1.
crossover	The crossover rate: must be between 0 and 1.
maxW	The maximum weight allowed by the random generator on the initialisation of the weight. During the mutation, weight can excede this boundry.

`minW` The minimum weight allowed by the random generator on the initialisation of the weight. During the mutation, weight can excede this boundry.

`maxGen` The maximum generation allowed if the error criterion is not reached.

`error` Error criterion to be reached. Desired mean squared error.

`printBestChromosone` For a better understanding, this option prints the weight of the best chromosome for each generation.

`...`

Note

Once an object of class "ANN" has been created, you can `print()` and `plot()` to visualise information about that object.

Author(s)

Francis Roy-Desrosiesrs

Examples

```
#use the command demo(ANN) to see how dataANN was created
data("dataANN")

#Example
        ANNGA(x    =input,
                y       =output,
                design  =c(1, 3, 1),
                population  =100,
                mutation = 0.3,
                crossover = 0.7,
                maxGen  =1000,
                error   =0.001)


#More examples
demo(ANN)
```

---

demoResultANN *Result of the demo*

Description

List of objects created by the demo. Increases accessibilty to results by eliminating computation step.

Usage

```
demoResultANN
```

Examples

```
data("dataANN")
demoResultANN
```

`predict.ANN` *Predict the output with an ANN already optimised.*

Description

Once the Artificial Neural Network is optimized, the output can be predicted.

Usage

```
## S3 method for class 'ANN'
predict(object, input,...)
```

Arguments

`object`	Object of class "ANN", an already optimized ANN.
`input`	Input inserted into the ANN to predict outputs.
`...`	not used

Author(s)

Francis Roy-Desrosiesrs

Examples

```
#use the command demo(ANN) to see how dataANN was created
data("dataANN")

par( mfrow=c(2,2) )
                plot(sin(t), main="sin")
                plot(noisy_sin, main="sin + 0.5 * rnorm")
                plot(output, main="sin + 0.5 * rnorm in [0,1]")
                plot(input, main="input should be in the range [0,1]")


demoResultANN

ANNObject<-demoResultANN[[3]]
```

```
ANNObject  #or print(ANNObject)
plot(ANNObject)


p <-predict( ANNObject, input )
p
plot(p$predict,main="Predicted Output")
```

Index